3-14-00

03/13/00 jc780 U.S. PTO

PATENT APPLICATION

jc525 U.S. PTO 09/524172 03/13/00

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

UTILITY PATENT APPLICATION TRANSMITTAL LETTER

Atty./Agent Docket No.: CR00219M

Mailing Date: March 13, 2000

Express Mail Label No.: EJ923752232US

Assistant Commissioner for Patents
Box Patent Application
Washington, D.C. 20231

Dear Sir:

Transmitted herewith for filing under 37 CFR 1.53 (b) is a Non-provisional Utility Patent:

_X_ New Application; or a ____ Continuation, ____ Division, or _ Continuation-in-Part (CIP) Application of prior US application No./____, filed on ____, having US Examiner ____, in Group Art Unit ____: of

Inventor(s): Brian Keith Classon
Vipul Anil Desai

For (Title): DECODER-USABLE SYNDROME GENERATION WITH REPRESENTATION GENERATED WITH INFORMATION BASED ON VECTOR PORTION

This transmittal letter has _2_ total pages.

Enclosed are:

_X_ _6_ sheets of drawings, along with _31_ pages of specification, claims, and abstract.

_X_ Oath or Declaration Combined with Power of Attorney (_3_ pages)
- _X_ Newly Executed (original or copy)
- ___ Copy from a prior application (if this is a Continuation/Division with no new matter)
- ___ Statement deleting named inventor(s) in prior application if this is a Continuation/Division (See 37 CFR 1.63(d)(2) and 1.33(b).)
- ___ Consider as the above Statement, Please delete as inventors for this application the following inventors named in the prior application: ____.

___ A certified copy of a ____ (non-US) application S/N __/____, having a filing date of ____, and foreign priority to this non-US application for the present application is hereby claimed under 35 USC 119.

_X_ An Assignment Transmittal Letter and Assignment of the invention to MOTOROLA, INC.

_X_ An Information Disclosure Statement (IDS), with PTO-1449, and _6_ citation copies.

___ Preliminary Amendment

___ Petition For Extension of Time for parent application of the present Continuation/Division/CIP application

_X_ PrintEFS Document

09524172 .031300

Atty./Agent Docket No.: CR00219M-Classon et al.

X Return Receipt Postcard

Instructions:

______ Incorporation by Reference (for Continuation/Division application) The entire disclosure of the prior application, from which a copy of the oath or declaration is supplied, is considered as being part of the disclosure of the accompanying application and is hereby incorporated by reference therein.

______ Since the present application is based on a prior US application, please amend the specification by adding the following sentence before the first sentence of the specification: "The present application is based on prior US application No. __________/________, filed on__________, which is hereby incorporated by reference, and priority thereto for common subject matter is hereby claimed."

______ Please cancel filed claims ________________.

X The filing fee is calculated as follows:

CLAIMS AS FILED, LESS ANY CANCELED BY AMENDMENT

	NUMBER OF CLAIMS	NUMBER EXTRA	RATE	FEE
TOTAL CLAIMS	39- 20 =	19	X $18	= $342.00
INDEPENDENT CLAIMS	3- 3 =	0	X $78	= $0.00
	MULTIPLE DEPENDENT CLAIMS		$260	= $ 0.00
			BASIC FEE	= $ 690.00
			TOTAL FILING FEE	= $1,032.00

X Please charge Deposit Account No. 13-4772 in the amount of $ 1,032.00 for the Total Filing Fee. Two duplicate copies of this sheet are enclosed.

X The Commissioner is hereby authorized to charge any additional fees which may be required now or in the future under 37 CFR 1.16 or 37 CFR 1.17, including any present or future time extension fees which may be required, or credit any overpayment to Deposit Account No. 13-4772. Two duplicate copies of this sheet are enclosed.

Please forward all correspondence to:

Customer Number **22917**

ES923752232US
"Express Mail" __________

Date of Deposit 3/13/00

I hereby certify that this paper or fee is being deposited with the United States Postal Service "Express Mail Post Office to Addressee" service under 37CFR 1.10 on the date indicated above and is addressed to the Assistant Commissioner for Patents, Washington, D.C. 20231.

KAREN KASS
Name of person mailing paper or fee

Karen Kass 3/13/00
Signature Date

By: Douglas D. Fekete

Douglas D. Fekete
Registration No. 29,065
MOTOROLA, INC.
Phone: (847) 538-2447
Fax: (847) 576-3750

INVENTOR INFORMATION

Inventor One Given Name:: Brian K
Family Name:: Classon
Postal Address Line One:: 124 Red Cedar Drive
City:: Streamwood
State or Province:: IL
Country:: USA
Postal or Zip Code:: 60107
City of Residence:: Streamwood
State or Province of Residence:: IL
Country of Residence:: USA
Citizenship Country:: 60107
Inventor Two Given Name:: Vipul A
Family Name:: Desai
Postal Address Line One:: 1480 Hassell Road
City:: Hoffman Estates
State or Province:: IL
Country:: USA
Postal or Zip Code:: 60195
City of Residence:: Hoffman Estates
State or Province of Residence:: IL
Country of Residence:: USA
Citizenship Country:: 60195

CORRESPONDENCE INFORMATION

Correspondence Customer Number:: 22917
Fax One:: 847-576-3750

APPLICATION INFORMATION

Title Line One:: DECODER-USABLE SYNDROME GENERATION WITH
Title Line Two:: REPRESENTATION GENERATED WITH INFORMATIO
Title Line Three:: N BASED ON VECTOR PORTION
Total Drawing Sheets:: 6
Formal Drawings?:: No
Application Type:: Utility
Docket Number:: CR00219M
Secrecy Order in Parent Appl.?:: No

Source:: PrintEFS Version 1.0.1

DECODER-USABLE SYNDROME GENERATION WITH REPRESENTATION GENERATED WITH INFORMATION BASED ON VECTOR PORTION

FIELD OF THE INVENTION

The present invention relates generally to decoders and more particularly to a syndrome usable in a decoder.

BACKGROUND OF THE INVENTION

As will be understood by those skilled in the art, a syndrome comprises a basic element of a decoding procedure. In one example, the syndrome can be used to identify the bits in error.

One example of a decoder comprises a Bose, Chaudhuri, and Hocquenghem ("BCH") decoder. A BCH decoder operates with BCH codes. BCH codes constitute a broad class of error-correcting and error-detecting codes. BCH codes are typically designed using the theory developed for Galois field ("GF") arithmetic. Examples of BCH codes include binary BCH codes and Reed-Solomon codes. Hamming codes, for example, comprise a subset of binary BCH codes. BCH codes comprise a subset of cyclic codes, as will be understood by those skilled in the art.

A binary BCH code, in one example, is characterized by a block length N, an information vector length K, a number of bits per symbol m where $N = 2^m-1$, a primitive element α used to construct a Galois field GF(2^m), a generator polynomial $g(x)$, and a guaranteed error correction capability t. Commonly, those skilled in the art interchangeably employ the terms "vector" and "polynomial." For example, the K-bit information vector $u[0],\ldots,u[K-1]$ is represented as the polynomial

$u(x) = u[K-1]x^{K-1} + u[K-2]x^{K-2} + \ldots + u[1]x + u[0]$, where the coefficients $u[0]$, ..., and $u[K-1]$ are elements of GF(2^m), and are binary in the case of binary BCH codes.

A detailed example is now presented for explanatory purposes. A K-bit information vector $u(x)$ is encoded into an N-bit codeword $v(x)$ by polynomial multiplication with $g(x)$. In one example, the encoder is initialized to some value, such as zero, before the polynomial multiplication. This codeword is transmitted, possibly corrupted by the channel, and received as a vector $r(x)$. A BCH decoder estimates a decoded information vector $w(x)$ based on $r(x)$ and possibly additional information.

Several decoder techniques have been developed to estimate $w(x)$ from $r(x)$. In one example of a BCH decoder, syndromes are computed by evaluating the received vector $r(x)$ at consecutive roots of the generator polynomial $g(x)$. The syndromes characterize the difference from the received vector to the nearest codeword. So, the syndromes can be used in a decoder to correct this difference.

The properties of a generator polynomial, in one example, can suggest a design of a BCH decoder. One can assume, for example, that the necessary roots of the generator polynomial comprise at least $2t$ consecutive values α^{L+1}, α^{L+2}, ..., α^{L+2t}, where α represents the primitive element used to construct the Galois field GF(2^m), and L represents a starting power of the roots. In one example, when the roots comprise α^1, α^2, ..., α^{2t} (L=0), it is understood by those skilled in the art that the syndromes s_j for binary BCH codes possess a property as described by the following exemplary Equation (1).

$$s_{2j} = (s_j)^2 \quad j = 1, \ldots, t \tag{1}$$

Because of Equation (1), in one example, the odd-numbered syndromes can be first computed by evaluating the received vector $r(x)$ at the necessary odd-powered roots of the generator polynomial. The even-numbered syndromes can then computed by using Equation (1). In this example, the number of operations in a BCH decoder can be reduced.

In another example, when the necessary roots of the generator polynomial are consecutive, the first root and error-correcting capability t comprise sufficient information to determine the subsequent roots of the generator polynomial. In this example, a BCH decoder can reduce the amount of information (e.g., memory usage) needed.

Computing syndromes typically comprises a computationally-intensive task because the computation typically involves polynomial evaluation using Galois field arithmetic. To compute syndrome s_j directly, the received vector $r(x)$ is evaluated at root α^{j+L}, as follows in exemplary Equation (2).

$$s_j = r\left(\alpha^{j+L}\right) = r_{N-1}\left(\alpha^{j+L}\right)^{N-1} + r_{N-2}\left(\alpha^{j+L}\right)^{N-2} + \ldots + r_1\left(\alpha^{j+L}\right) + r_0 \quad (2)$$

Evaluating Equation (2), in one example, requires N-1 GF(2^m) additions, N-1 general GF(2^m) multiplications, and N-1 general GF(2^m) exponentiations per syndrome.

When representing the Galois field elements as an m-tuple over a standard canonical basis $(\alpha^{m-1},\ldots,\alpha,1)$, GF($2^m$) addition is equivalent to a simple exclusive-OR operation. Further, GF(2) multiplication is equivalent to a logical AND operation for $m = 1$. However, the general GF(2^m) multiplications and general GF(2^m) exponentiations typically have no simple implementation for $m > 1$ in the standard canonical basis. To minimize the number of these exponentiations, one technique

for evaluation of polynomials employs an iterative algorithm, such as Horner's rule. For instance, Equation (2) can be expressed as follows in exemplary Equation (3).

$$s_j = r(\alpha^{j+L}) = r(\beta) = (\ldots((r_{N-1}\beta + r_{N-2})\beta + r_{N-3})\beta + \ldots + r_1)\beta + r_0 \text{ where } \beta = \alpha^{j+L} \quad (3)$$

On the one hand, Equation (3) eliminates the general GF(2^m) exponentiations. On the other hand, Equation (3) nevertheless as a shortcoming requires N-1 general GF(2^m) multiplications per syndrome.

To address the shortcoming of requiring GF(2^m) multiplications in syndrome calculation, a number of hardware and software approaches have been offered. For example, certain hardware designs implement a general GF(2^m) multiplier. One shortcoming of a typical hardware implementation of a general GF(2^m) multiplier is the relatively large amount of power consumption required by the general GF(2^m) multiplier.

Further, a number of software implementation that use lookup tables have been offered to perform general GF(2^m) multiplication for syndrome calculations. In one example, the multiplicands are transformed from the standard canonical basis into an exponential representation by using lookup tables. In the exponential representation, integer arithmetic (i.e., addition, modulo) replaces GF(2^m) multiplication. Once the integer result has been computed, a lookup table is used to transform the integer result into the standard canonical basis. For instance, if a and b are elements of GF(2^m), the steps of a general GF(2^m) multiplier in one software implementation for computing $c = a \times b$ are as follows.

$$
\begin{aligned}
& if\ ((a == 0) or (b == 0)) \\
& \quad c = 0 \\
& else \\
& \quad a' = \mathrm{LUT}(a) \\
& \quad b' = \mathrm{LUT}(b) \\
& \quad c' = (a' + b') \bmod (2^m - 1) \\
& \quad c = \mathrm{LUT}^{-1}(c')
\end{aligned}
$$

In the above steps, LUT is the lookup table used to transform a number from the standard canonical basis into an exponential representation, and LUT^{-1} is the lookup table that transforms an exponential representation into the standard canonical basis.

This exemplary software implementation suffers from the use of lookup tables. As one shortcoming, the memory requirement of the lookup tables is proportional to 2^m. For modest m, such as m=9, the memory requirement therefore undesirably exceeds 512 words. As another shortcoming, the required accessing of the lookup tables limits the processing capabilities of the software implementation. This use of the lookup table undesirably lowers the data rate that is sustainable for a software implementation.

Thus, a need exists for enhanced generation of a syndrome that is usable in a decoder.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. 1 is a functional block diagram of one example of a communications system that encodes and decodes a vector.

FIG. 2 is a functional block diagram that depicts exemplary details of a decoder component of the communications system of FIG. 1.

FIG. 3 is a functional block diagram that depicts illustrative details of one example of a syndrome generator component of the decoder component of FIG. 2.

FIG. 4 is a functional block diagram of exemplary details of a reducer component of the syndrome generator component of FIG. 3.

FIG. 5 is a functional block diagram of exemplary details of a linear feedback shift register element component of the reducer component of FIG. 4.

FIG. 6 depicts one example of logic employed by a converter component of the syndrome generator component of FIG. 3.

FIG. 7 depicts exemplary logic employed by one example of a syndrome computer component of the decoder component of FIG. 2.

DETAILED DESCRIPTION OF THE INVENTION

The invention encompasses a method for generating a syndrome usable in a decoder. There is employed information, that is based on a portion of a vector, to generate a representation. There is generated, with employment of the representation, the syndrome.

Another embodiment of the invention encompasses a system for generating a syndrome usable in a decoder. A reducer employs information, that is based on a portion of a vector, to generate a representation. A converter generates, with employment of the representation, the syndrome.

A further embodiment of the invention encompasses an article of manufacture. At least one computer usable medium has computer readable program code means embodied therein for causing generation of a syndrome usable in a decoder. There is provided computer readable program code means for causing a computer to employ information, that is based on a portion of a vector, to generate a representation. There is also provided computer readable program code means for causing a computer to generate, with employment of the representation, the syndrome.

In one advantageous aspect, the invention can enable lowered-complexity correction of a number of errors, for example, bit errors. For instance, this can serve to desirably increase life of a battery. In one example, the invention serves to reduce the overall complexity of error correction by advantageously reducing the complexity of syndrome calculation.

A detailed discussion of an exemplary embodiment of the invention is presented herein, for illustrative purposes.

FIG. 1 illustrates a functional block diagram of one example of a communications system 100. System 100, in one example, includes a plurality of components such as computer software and/or hardware components. For instance, a number of such components can be combined or divided, as will be appreciated by those skilled in the art.

Referring still to FIG. 1, in one example of system 100, an encoder 110 is initialized with an initialization 118. The encoder 110, for instance, multiplies an information vector $u(x)$ 112 by a generator polynomial $g(x)$ 114 to produce a codeword vector $v(x)$ 116. In one example, the necessary roots of the generator polynomial $g(x)$ 114 comprise at least $2t$ consecutive values. For example, t comprises a guaranteed error correction capability of a binary Bose, Chaudhuri, and Hocquenghem ("BCH") code. The codeword vector $v(x)$ 116 is transmitted through a channel 120. In one example, the channel 120 possibly corrupts the codeword vector $v(x)$ 116. The output of the channel 120 is a received vector $r(x)$ 122. A decoder 130 processes the received vector $r(x)$ 122 by using the generator polynomial $g(x)$ 114 to produce a decoded vector $w(x)$ 132. The decoded vector $w(x)$ 132 comprises an estimate of the information vector $u(x)$ 112 by the decoder 130. One example of decoder 130 comprises (e.g., computer) processor 602 coupled with memory 604. Memory 604, in one example, serves to store logic, for instance, a software implementation.

FIG. 2 illustrates exemplary details of the decoder 130 of the communications system 100 (FIG. 1). In one example, a roots ROM 210 comprises the necessary roots of the generator polynomial $g(x)$ 114 and produces a roots array $\underline{\alpha}$ 215. In one example, the necessary roots can be computed from the polynomial $g(x)$ 114. In another example, the roots of the polynomial $g(x)$ 114 can be precomputed by the

decoder 130 and stored in the roots ROM 210. In a further example, the roots ROM 210 can comprise at least one root when the necessary roots are sequential. In such an example, the first root of the sequence and the guaranteed error correction capability are sufficient. In yet another example, the roots ROM 210 can comprise a particular subset of the necessary roots. In such an example with the starting power L=0, the odd-powered roots, such as α, α^3, α^5, etc., comprise the subset.

Again referring to FIG. 2, a syndrome computer 240 of decoder 130, in one example, comprises a plurality of syndrome generators 220 for processing the received vector $r(x)$ 122 and the roots array $\underline{\alpha}$□215 to produce a plurality of syndromes $\underline{s}$ 225. For instance, a j^{th} instance of syndrome generator 220 processes a generator polynomial root 212, which corresponds to a j^{th} element of the roots array $\underline{\alpha}$ 215, and the received vector $r(x)$ 122 to produce a j^{th} instance of syndrome s_j 222. For example, a syndrome processor 230 processes the plurality of syndromes $\underline{s}$ 225, the generator polynomial $g(x)$ 114, and the received vector $r(x)$ 122 to produce the decoded vector $w(x)$ 132.

Further referring to FIG. 2, in one example, syndrome computer 240 comprises multiple parallel syndrome generators 220. In another example, syndrome computer 240 comprises a single syndrome generator 220. For instance, the single syndrome generator 220 can be employed with additional hardware (not shown), as will be understood by those skilled in the art.

FIG. 3 illustrates exemplary details of the syndrome generator 220 of the decoder 130 (FIGS. 1-2). In one example, a reducer 340 processes the received vector $r(x)$ 122 using the initialization 118 and a minimal polynomial $p_j(x)$ 352 for producing a representation $c_j(x)$ 342. The minimal polynomial $p_j(x)$ 352 corresponds

to the generator polynomial root 212. For instance, a reduction mask ROM 350 comprises the minimal polynomial $p_j(x)$ 352. In a further example, the reduction mask ROM 350 also comprises a minimal polynomial degree k_j 354 corresponding to the minimal polynomial $p_j(x)$ 352. In one example, the minimal polynomial $p_j(x)$ 352 and the minimal polynomial degree k_j 354 can be precomputed because the decoder 130 knows the generator polynomial $g(x)$ 114 (FIGS. 1-2). A detailed discussion of an exemplary procedure for performing such a computation is presented herein. In one example, reduction mask ROM 350 comprises one or more reduction masks 351. For instance, syndrome generator 220 generates a reduction masks 351 from a generator polynomial root 212, and employs the reduction mask 351 to generate representation $c_j(x)$ 342.

Again referring to FIG. 3, a conversion mask ROM 320 of syndrome generator 220 comprises one or more instances of conversion mask $d_j(x)$ 322 corresponding to the generator polynomial root 212 and the minimal polynomial degree k_j 354. For example, the one or more conversion masks $d_j(x)$ 322 can be precomputed because the decoder 130 (FIGS. 1-2) knows the generator polynomial $g(x)$ 114 (FIGS. 1-2). In one example, a converter 330 employs (e.g., transforms) the representation $c_j(x)$ 342 using the one or more conversion masks $d_j(x)$ 322 to generate (e.g., produce) the syndrome s_j 222.

FIG. 4 illustrates exemplary details of the reducer 340 of the syndrome generator 220 (FIGS. 2-3). In one example, the reducer 340 comprises a linear feedback shift register ("LFSR") 480 and an indexer 430. For example, the indexer 430 is responsive to the minimal polynomial degree k_j 354 for producing an index q_0 431 set to 0, an index q_i 432 set to i-1, etc., and an index $q_{k\text{-}1}$ 433 set to k_j-1. The LFSR 480 comprises, for instance, a number of LFSR elements 490. The index q_i

432, in one example, can control the inputs to the LFSR element 490. In one example, the number of LFSR elements 490, is determined by the minimal polynomial degree k_j 354. LFSR elements 490 produce, for example, outputs c_0 441, c_i 442, etc., and c_{k-1} 443 that comprise the representation $c_j(x)$ 342. In addition, LFSR elements 490 produce symbols f_0 421, f_i 422, f_{i+1} 423, etc., and f_{k-1} 420. In one example, f_{k-1} 420 comprises a feedback symbol for recursion with respect to the LFSR elements 490.

FIG. 5 illustrates exemplary details of an LFSR element 490 of the LFSR 480 (FIG. 4). For instance, a switch 550 selects an output 532, from a GF(2) adder 530, to produce a delay input 552. At reset, the switch 550 selects an extracted initialization 514. A delay 520 holds the delay input 552 from the switch 550, and produces a delay output 522. A switch 560 directs the delay output 522 to the symbol f_{i+1} 423. The switch 560, after processing information (e.g., output 522) that is based on a portion of a received vector $r(x)$ 122 (FIGS. 2-4), directs the delay output 522 to the output c_i 442.

Further referring to FIG. 5, an extractor 513 of LFSR element 490, in one example, uses the index q_i 432 to extract the corresponding element of initialization 118, for producing the extracted initialization 514. For instance, an extractor 511 uses the index q_i 432 to extract the corresponding element of the minimal polynomial $p_j(x)$ 352, for producing a binary coefficient p 512. For example, a GF(2) multiplier 540 multiplies the binary coefficient p 512 and the feedback symbol f_{k-1} 420 to produce a product symbol 542. The multiplier 540, in one example, operates over GF(2), so its operation advantageously reduces to a logical-AND operation. For instance, GF(2) adder 530 combines the symbol f_i 422 and the product symbol 542, to produce the output 532.

Referring now to FIG. 3, an exemplary procedure for computing the minimal polynomial $p_j(x)$ 352 and its minimal polynomial degree k_j 354 corresponding to the generator polynomial root 212, is discussed. One can assume, for example, that the necessary roots of the generator polynomial $g(x)$ 114 (FIGS. 1-2) comprise the values α^{L+1}, α^{L+2}, ..., α^{L+2t}. The assumption of consecutive roots is reasonable because most generator polynomials have consecutive roots. Although the starting power L can be set to an arbitrary value, L is often set to 0.

For instance, the minimal polynomial $p_j(x)$ 352 is computed by employing the following exemplary Equations 4-5.

$$p_j(x) = \prod_{i \in CC(j)} (x - \alpha^i) \tag{4}$$

$$CC(j) = \{(j \cdot 2^l) \bmod (2^m - 1), l = 0,1,2\ldots,k_j - 1\} \tag{5}$$

In Equations 4-5, m is the number of bits per symbol, α is the primitive element used to construct the Galois field GF(2^m), k_j (the minimal polynomial degree k_j 354) is the degree of the minimal polynomial $p_j(x)$ 352 with $k_j \leq m$, and CC(j) is a cyclotomic coset of α_j.

In addition, the following illustrative Table 1 lists the minimal polynomial degrees k_j 354 of a given generator polynomial root 212 and a given m for BCH codes of length $\leq 2^{13}$-1 that correct up to t=7 errors.

Table 1: Minimal polynomial degrees k_j 354.

M	2^m-1	α	α^3	α^5	α^7	α^9	α^{11}	α^{13}
3	7	3	3	α^3	N/A	N/A	N/A	N/A
4	15=3×5	4	4	<u>2</u>	4	α^3	α^7	α^7
5	31	5	5	5	5	α^5	5	α^{11}
6	63=3×3×7	6	6	6	6	<u>3</u>	6	6
7	127	7	7	7	7	7	7	7

M	2^m-1	α	α^3	α^5	α^7	α^9	α^{11}	α^{13}
8	255=3×5×17	8	8	8	8	8	8	8
9	511=7×73	9	9	9	9	9	9	9
10	1023=3×11×31	10	10	10	10	10	10	10
11	2047=13×89	11	11	11	11	11	11	11
12	4095=3×3×5×7×13	12	12	12	12	12	12	12
13	8191	13	13	13	13	13	13	13

For a number of codes of interest, the minimal polynomial degree k_j 354 equals m for all syndromes $\underline{s}$ 225 that need to be computed. If a polynomial root has the same minimal polynomial $p_j(x)$ 352 as another root, the other root is listed instead of the degree. Note that the underlined entries in Table 1 have $k<m$: α^5 in GF(2^4) and α^9 in GF(2^6). Also note that the degree of all but the two entries with $k<m$ and of α^9 in GF(2^4), can be computed as follows.

1. m prime $\Rightarrow k=m$
2. If j is the exponent of the element α_j, when $\gcd(2^m-1,j)=1 \Rightarrow k=m$ ("gcd" represents the greatest common denominator operation).
3. If j is the exponent of the element α_j, $m>2\lfloor \log_2 j \rfloor \Rightarrow k=m$ where

$$\lfloor x \rfloor = \begin{cases} x & x \text{ integer} \\ \operatorname{int}(x+1) & \text{otherwise} \end{cases}$$

For the exceptions, a table of published minimal polynomials can be used, or the cyclotomic cosets can be computed using Equation (5).

In one example, use of minimal polynomial $p_j(x)$ 352 advantageously allows the general GF(2^m) multiplier employed in previous designs, to be replaced by reducer 340 and the converter 330, as in FIG. 3. The reducer 340 desirably operates in a k-tuple basis with respect to $(\alpha^j)^{k-1},\ldots,\alpha^j,1$, whereas a general GF($2^m$) multiplier operates in the standard canonical basis $(\alpha^{m-1},\ldots,\alpha,1)$. The converter 330 advantageously transforms the k-tuple basis into the standard canonical basis. The

k-tuple basis with respect to $(\alpha^j)^{k-1},\ldots,\alpha^j,1$ desirably avoids the previous need for the general GF(2^m) multiplier, by advantageously transforming the syndrome evaluation of Equation (3) into the representation evaluation of the following exemplary Equation (6).

$$c_j = r(\alpha') = (\ldots((r_{N-1}\alpha' + r_{N-2})\alpha' + r_{N-3})\alpha' + \ldots + r_1)\alpha' + r_0 \tag{6}$$

In Equation (6), α' is a primitive element of GF(2^k). Those skilled in the art will appreciate that evaluation of a polynomial by a primitive element (such as evaluation of $r(x)$ 122 by the primitive element α' in GF(2^k)) can be performed by an LFSR 480. In a typical example of LFSR 480, all operations involve binary operands; for example, multiplication of binary operands is equivalent to a logical-AND operation.

Now referring to FIG. 3, in one example, one or more instances of conversion mask $d_j(x)$ 322 of the conversion mask ROM 320, are computed from the generator polynomial root 212 and the minimal polynomial degree k_j 354 such as by the following exemplary Equation (7).

$$d_i = (\alpha^j)^i \quad i = 0,1,\ldots,k_j - 1 \tag{7}$$

In Equation (7), conversion mask ROM 320 comprises k_j conversion masks $d_j(x)$ 322 that each comprise one or more bits.

Again referring to FIG. 3, the converter 330, in a further example, is responsive to one or more instances of conversion mask $d_j(x)$ 322 and representation $c_j(x)$ 342, for computing the syndrome s_j 222 such as by the following exemplary Equation (8).

$$s_j = c_{k-1}d_{k-1} + \ldots + c_1d_1 + c_0d_0 = c_{k-1}d_{k-1} + \ldots + c_1d_1 + c_0 \tag{8}$$

In Equation (8), the members $c[0]$, ..., and $c[k\text{-}1]$ of the representation $c_j(x)$ 342 are binary, as a result of using the k-tuple basis with respect to $(\alpha^j)^{k-1},\ldots,\alpha^j,1$. Although Equation (8) may appear to require general GF multiplications, advantageously no general GF(2^m) multiplication is necessary. Because the members $c[0]$, ..., and $c[k\text{-}1]$ are binary, a desirably simple algorithm such as

$$
\begin{aligned}
& z = c_0 \\
& \text{do } i = 1,\ldots,k_j - 1 \\
& \quad z = z\,\hat{}\begin{cases} d_i & \text{if } c_i = 1 \\ 0 & \text{otherwise} \end{cases} \\
& \text{end do} \\
& s_j = z
\end{aligned}
$$

can be used by converter 330 for the conversion, where z comprises an adder. Furthermore, a number of other implementations of Equation (8) may be employed, as will be appreciated by those skilled in the art.

FIG. 6 represents one example of logic 600 for producing syndromes $\underline{s}$ 225 (FIG. 2). In one example, converter 330 (FIG. 3) employs (e.g., performs) logic 600. Initialization 610 sets a counter i to 0. Initialization 615 sets adder z to 0. STEP 620 fetches the i^{th} element of one or more instances of conversion mask $d_j(x)$ 322. In STEP 625, the fetched element from STEP 620, is added (in the GF(2^m) sense) to the adder z if the i^{th} element of the representation $c_j(x)$ 342 is 1. STEP 630 increments counter i by 1. DECISION 635 checks to determine whether the counter i (as present before STEP 630) is less than minimal polynomial degree k_j 354. STEP 640 assigns the contents of the adder z (as computed in STEP 625) to the syndrome s_j 222.

FIG. 7 represents one example of logic 700 for producing syndromes $\underline{s}$ 225 (FIG. 2). In one example, syndrome computer 240 (FIG. 2) employs (e.g., performs) logic 700. A further example employs a starting power L=0. From Equation (2), a result of setting L=0 is that a syndrome s_j 222 corresponds to a generator polynomial root α^j 212. So, in one example, an odd-numbered syndrome s_{2j-1} 222 corresponds to an odd-powered generator polynomial root α^{2j-1} 212. Further, the odd-numbered syndrome s_{2j-1} 222 has an odd-numbered representation $c_{2j-1}(x)$ 342. Similarly, an even-numbered syndrome s_{2j} 222 corresponds to an even-powered generator polynomial root α^{-2j} 212. In addition, the even-numbered syndrome s_{2j} 222 has an even-numbered representation $c_{2j}(x)$ 342.

In one example, any even number e can be expressed as a product of an odd integer b and powers of two, such as by employment of the following exemplary Equation (9).

$$b = e/2^h \qquad (9)$$

In Equation (9), the power h is a non-negative integer.

From Equations (1) and (9), those skilled in the art will appreciate that an even-numbered syndrome s_e 222 can be computed by employing an odd-numbered representation $c_b(x)$ 342 and one or more instances of conversion mask $d_e(x)$ 322, such as by employment of the following exemplary Equation (10).

$$s_e = \sum_{i=0}^{k_j-1} c_i^{(b)} d_i^{(e)} \qquad (10)$$

In Equation (10), $c_i^{(b)}$ comprises the i^{th} member of the odd-numbered representation $c_b(x)$ 342. In addition, $d_i^{(e)}$ comprises the i^{th} member of the one or more instances of

conversion mask $d_e(x)$ 322. In one example, $d_i^{(e)}$ of Equation (10) comprises a modified version of Equation (8), as will be understood by those skilled in the art. In one advantageous aspect, Equation (10) provides evaluation of even-numbered syndromes s_e 222 with desirable avoidance of, for example, the exponentiation of Equation (2), as will be appreciated by those skilled in the art.

Now referring to FIG. 7, initialization 710 sets a counter j to 1. DECISION 715 checks to determine whether the counter j is even or odd. In one example, DECISION 715 determines whether the generator polynomial root 212 is even-powered or odd-powered. When DECISION 715 determines that the counter j is odd, STEP 720 processes received vector $r(x)$ 122 using initialization 118 and minimal polynomial $p_j(x)$ 352, to produce an odd-numbered instance of representation $c_j(x)$ 342. Since L=0, in an example, the minimal polynomial $p_j(x)$ 352 corresponds to an odd-powered root of generator polynomial $g(x)$ 114. STEP 725 stores the odd-numbered instance of representation $c_j(x)$ 342, for example, in random access memory ("RAM") 726.

Referring again to FIG. 7, STEP 730, in one example, employs logic 600 (FIG. 6) to produce an odd-numbered instance of syndrome s_j 222 from an odd-numbered instance of representation $c_j(x)$ 342 , as will be appreciated by those skilled in the art. STEP 735 increments counter j by 1. DECISION 740 determines whether the counter j (as present after STEP 735) is less than or equal to the number, y, of necessary roots. In one example, y comprises $2t$ values.

Further referring to FIG. 7, in the event that DECISION 715 determines counter j is even, STEP 750 factors the counter j into an odd number b and powers of two. One example of such factorization can employ Equation (9). STEP 755, in

one example, fetches from RAM 726 the odd-numbered instance of representation $c_j(x)$ 342 corresponding to the odd-number b. STEP 760, in one example, employs logic 600 (FIG. 6) to produce an even-numbered instance of syndrome s_j 222 from an odd-numbered instance of representation $c_j(x)$ 342 (determined in STEP 755).

Referring again to FIG. 7, logic 700, in one example, serves to produce syndromes $\underline{s}$ 225 (FIG. 2) based on sequential processing of the syndromes $\underline{s}$ 225. In another example, logic 700 first computes instances of odd-numbered representation $c_{2j-1}(x)$ 342. Subsequently, in this example, logic 700 computes syndromes $\underline{s}$ 225, in any order, from the odd-numbered instances of representation $c_{2j-1}(x)$ 342. In yet another example, logic 700 produces syndromes $\underline{s}$ 225 by computing syndromes s_j 222 that correspond to an odd-numbered instance of representation $c_b(x)$ 342. With L=0 and t=4, one example of logic 700 can determine (e.g., compute) syndromes s_1, s_2, s_4, and s_8 once logic 700 has determined (e.g., computed) an odd-numbered instance of representation $c_1(x)$ 342, since syndromes s_1, s_2, s_4, and s_8 correspond to the same odd-numbered instance of representation $c_1(x)$ 342. Similarly, one example of logic 700 can determine syndromes s_3 and s_6 once logic 700 has determined an odd-numbered instance of representation $c_3(x)$ 342, since syndromes s_3 and s_6 correspond to the same odd-numbered instance of representation $c_3(x)$ 342, as will be appreciated by those skilled in the art.

The flow diagrams depicted herein are just exemplary. There may be many variations to these diagrams or the steps (or operations) described therein without departing from the spirit of the invention. For instance, the steps may be performed in a differing order, or steps may be added, deleted or modified. All these variations are considered a part of the claimed invention.

Although preferred embodiments have been depicted and described in detail herein, it will be apparent to those skilled in the relevant art that various modifications, additions, substitutions and the like can be made without departing from the spirit of the invention and these are therefore considered to be within the scope of the invention as defined in the following claims.

CLAIMS

What is claimed is:

1. A method for generating a syndrome usable in a decoder, the method comprising the steps of:

employing information, that is based on a portion of a vector, to generate a representation; and

generating, with employment of the representation, the syndrome.

2. The method of claim 1 wherein the step of employing the information, that is based on the portion of the vector, to generate the representation comprises the step of employing a number of minimal polynomials to operate on the portion of the vector.

3. The method of claim 2 wherein the step of employing the number of minimal polynomials to operate on the portion of the vector comprises the step of selecting the number of minimal polynomials to comprise a generator polynomial employed to encode the portion of the vector.

4. The method of claim 1 wherein the step of employing the information, that is based on the portion of the vector, to generate the representation and the step of generating, with employment of the representation, the syndrome comprise the step of generating a syndrome for a binary Bose-Chaudhuri-Hocquenghem code.

5. The method of claim 1 wherein the step of employing the information, that is based on the portion of the vector, to generate the representation and the step of generating, with employment of the representation, the syndrome comprise the step of generating a syndrome for a binary cyclic code.

6. The method of claim 1 wherein the step of generating, with employment of the representation, the syndrome comprises the step of converting and/or transforming the representation to obtain the syndrome.

7. The method of claim 1 wherein the step of employing the information, that is based on the portion of the vector, to generate the representation comprises the step of selecting the portion of the vector to comprise a portion of a preprocessed vector.

8. The method of claim 1 wherein the step of employing the information, that is based on the portion of the vector, to generate the representation comprises the steps of:

generating a reduction mask from a root of a generator polynomial; and

employing the reduction mask to generate the representation.

9. The method of claim 8 wherein the step of employing the reduction mask to generate the representation comprises the step of selecting the reduction mask to represent a minimal polynomial that is based on a cyclotomic coset and on the root of the generator polynomial.

10. The method of claim 8 wherein the representation comprises an odd-numbered representation, wherein the syndrome comprises an even-numbered syndrome, in combination with a method for generating the even-numbered syndrome, comprising the steps of:

employing the reduction mask to generate the odd-numbered representation; and

generating, with employment of the odd-numbered representation, the even-numbered syndrome.

11. The method of claim 10 wherein the root of the generator polynomial comprises an even-powered root of the generator polynomial, and wherein the step of generating, with employment of the odd-numbered representation, the even-numbered syndrome comprises the steps of:

determining a conversion mask from an even-powered root of the generator polynomial; and

converting, with employment of the conversion mask, the odd-numbered representation to obtain the even-numbered syndrome.

12. The method of claim 8 wherein the representation comprises an odd-numbered representation, wherein the syndrome comprises an odd-numbered syndrome, in combination with a method for generating the odd-numbered syndrome, comprising the steps of:

employing the reduction mask to generate the odd-numbered representation; and

generating, with employment of the odd-numbered representation, the odd-numbered syndrome.

13. The method of claim 1 wherein the step of generating, with employment of the representation, the syndrome comprises the steps of:

determining a conversion mask from a root of a generator polynomial; and

converting, with employment of the conversion mask, the representation to obtain the syndrome.

14. A system for generating a syndrome usable in a decoder, the system comprising:

a reducer that employs information, that is based on a portion of a vector, to generate a representation; and

a converter that generates, with employment of the representation, the syndrome.

15. The system of claim 14 wherein the reducer employs a number of minimal polynomials to operate on the portion of the vector.

16. The system of claim 15 wherein the reducer selects the number of minimal polynomials to comprise a generator polynomial employed to encode the portion of the vector.

17. The system of claim 14 wherein the reducer and the converter generate a syndrome for a binary Bose-Chaudhuri-Hocquenghem code.

18. The system of claim 14 wherein the reducer and the converter generate a syndrome for a binary cyclic code.

19. The system of claim 14 wherein the converter converts and/or transforms the representation to obtain the syndrome.

20. The system of claim 14 wherein the reducer selects the portion of the vector to comprise a portion of a preprocessed vector.

21. The system of claim 14 wherein the reducer employs a reduction mask to generate the representation, wherein the reduction mask is generated from a root of a generator polynomial.

22. The system of claim 21 wherein the reduction mask represents a minimal polynomial that is based on a cyclotomic coset and on the root of the generator polynomial.

23. The system of claim 21 wherein the representation comprises an odd-numbered representation, wherein the syndrome comprises an even-numbered syndrome, in combination with a system for generating the even-numbered syndrome, wherein the reducer employs the reduction mask to generate the odd-numbered representation, and wherein the converter generates, with employment of the odd-numbered representation, the even-numbered syndrome.

24. The system of claim 23 wherein the root of the generator polynomial comprises an even-powered root of the generator polynomial, and wherein the converter converts, with employment of a conversion mask, the odd-numbered representation to obtain the even-numbered syndrome, wherein the conversion mask is determined from an even-powered root of the generator polynomial.

25. The system of claim 21 wherein the representation comprises an odd-numbered representation, wherein the syndrome comprises an odd-numbered syndrome, in combination with a system for generating the odd-numbered syndrome, wherein the reducer employs the reduction mask to generate the odd-numbered representation, and wherein the converter generates, with employment of the odd-numbered representation, the odd-numbered syndrome.

26. The system of claim 14 wherein the converter converts, with employment of a conversion mask, the representation to obtain the syndrome, wherein the conversion mask is determined from a root of a generator polynomial.

27. An article of manufacture, comprising:

at least one computer usable medium having computer readable program code means embodied therein for causing generation of a syndrome usable in a decoder, the computer readable program code means in the article of manufacture comprising:

computer readable program code means for causing a computer to employ information, that is based on a portion of a vector, to generate a representation; and

computer readable program code means for causing a computer to generate, with employment of the representation, the syndrome.

28. The article of manufacture of claim 27 wherein the computer readable program code means for causing a computer to employ the information, that is based on the portion of the vector, to generate the representation comprises computer readable program code means for causing a computer to employ a number of minimal polynomials to operate on the portion of the vector.

29. The article of manufacture of claim 28 wherein the computer readable program code means for causing a computer to employ the number of minimal polynomials to operate on the portion of the vector comprises computer readable program code means for causing a computer to select the number of minimal polynomials to comprise a generator polynomial employed to encode the portion of the vector.

30. The article of manufacture of claim 27 wherein the computer readable program code means for causing a computer to employ the information, that is based on the portion of the vector, to generate the representation and the computer readable program code means for causing a computer to generate, with employment of the representation, the syndrome comprise computer readable program code means for causing a computer to generate a syndrome for a binary Bose-Chaudhuri-Hocquenghem code.

31. The article of manufacture of claim 27 wherein the computer readable program code means for causing a computer to employ the information, that is based on the portion of the vector, to generate the representation and the computer readable program code means for causing a computer to generate, with employment of the representation, the syndrome comprise computer readable program code means for causing a computer to generate a syndrome for a binary cyclic code.

32. The article of manufacture of claim 27 wherein the computer readable program code means for causing a computer to generate, with employment of the representation, the syndrome comprises computer readable program code means for causing a computer to convert and/or transform the representation to obtain the syndrome.

33. The article of manufacture of claim 27 wherein the computer readable program code means for causing a computer to employ the information, that is based on the portion of the vector, to generate the representation comprises computer readable program code means for causing a computer to select the portion of the vector to comprise a portion of a preprocessed vector.

34. The article of manufacture of claim 27 wherein the computer readable program code means for causing a computer to employ the information, that is based on the portion of the vector, to generate the representation comprises:

computer readable program code means for causing a computer to generate a reduction mask from a root of a generator polynomial; and

computer readable program code means for causing a computer to employ the reduction mask to generate the representation.

35. The article of manufacture of claim 34 wherein the computer readable program code means for causing a computer to employ the reduction mask to generate the representation comprises computer readable program code means for causing a computer to select the reduction mask to represent a minimal polynomial that is based on a cyclotomic coset and on the root of the generator polynomial.

36. The article of manufacture of claim 34 wherein the representation comprises an odd-numbered representation, wherein the syndrome comprises an even-numbered syndrome, wherein the at least one computer usable medium includes second computer readable program code means embodied therein for causing generation of the even-numbered syndrome, the second computer readable program code means in the article of manufacture comprising:

computer readable program code means for causing a computer to employ the reduction mask to generate the odd-numbered representation; and

computer readable program code means for causing a computer to generate, with employment of the odd-numbered representation, the even-numbered syndrome.

37. The article of manufacture of claim 36 wherein the root of the generator polynomial comprises an even-powered root of the generator polynomial, and wherein the computer readable program code means for causing a computer to generate, with employment of the odd-numbered representation, the even-numbered syndrome comprises:

computer readable program code means for causing a computer to determine a conversion mask from an even-powered root of the generator polynomial; and

computer readable program code means for causing a computer to convert, with employment of the conversion mask, the odd-numbered representation to obtain the even-numbered syndrome.

38. The article of manufacture of claim 34 wherein the representation comprises an odd-numbered representation, wherein the syndrome comprises an odd-numbered syndrome, wherein the at least one computer usable medium includes second computer readable program code means embodied therein for causing generation of the odd-numbered syndrome, the second computer readable program code means in the article of manufacture comprising:

computer readable program code means for causing a computer to employ the reduction mask to generate the odd-numbered representation; and

computer readable program code means for causing a computer to generate, with employment of the odd-numbered representation, the odd-numbered syndrome.

39. The article of manufacture of claim 27 wherein the computer readable program code means for causing a computer to generate, with employment of the representation, the syndrome comprises:

computer readable program code means for causing a computer to determine a conversion mask from a root of a generator polynomial; and

computer readable program code means for causing a computer to convert, with employment of the conversion mask, the representation to obtain the syndrome.

* * * * *

DECODER-USABLE SYNDROME GENERATION WITH REPRESENTATION GENERATED WITH INFORMATION BASED ON VECTOR PORTION

ABSTRACT OF THE DISCLOSURE

System (100) for generating syndrome (222) usable in decoder (130) includes reducer (340) and converter (330). Reducer (340) employs information, that is based on portion of vector (122), to generate representation (342). Converter (330) generates, with employment of representation (342), syndrome (222).

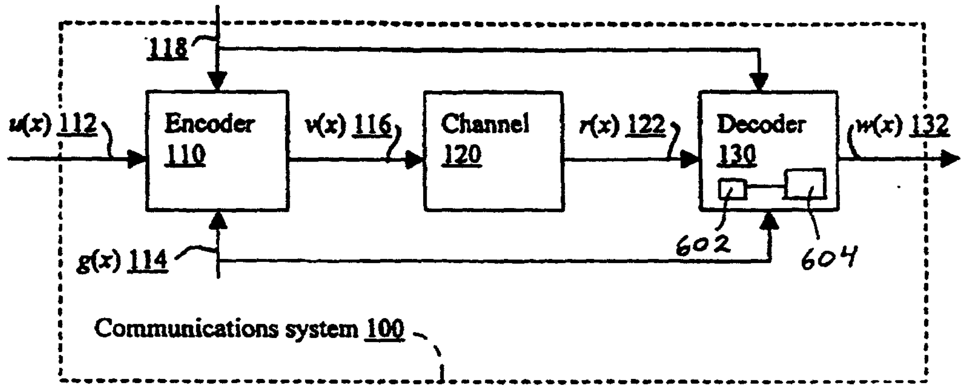

118
u(x) 112
Encoder
110
v(x) 116
Channel
120
r(x) 122
Decoder
130
w(x) 132
g(x) 114
602
604
Communications system 100

FIG. 1

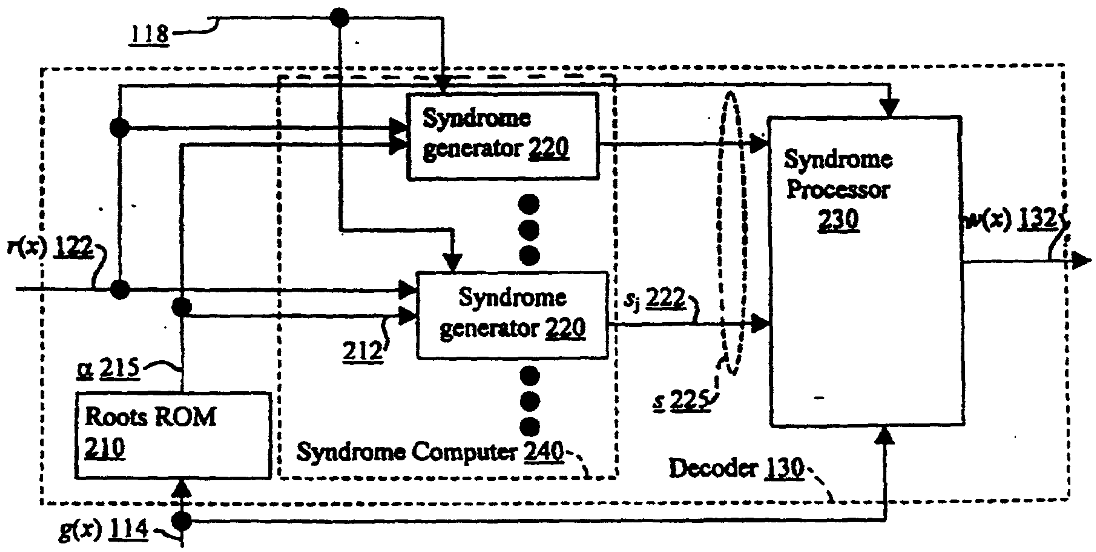

118
Syndrome
generator 220
Syndrome
Processor
230
r(x) 122
w(x) 132
Syndrome
generator 220
s_j 222
212
α 215
s 225
Roots ROM
210
Syndrome Computer 240
Decoder 130
g(x) 114

FIG. 2

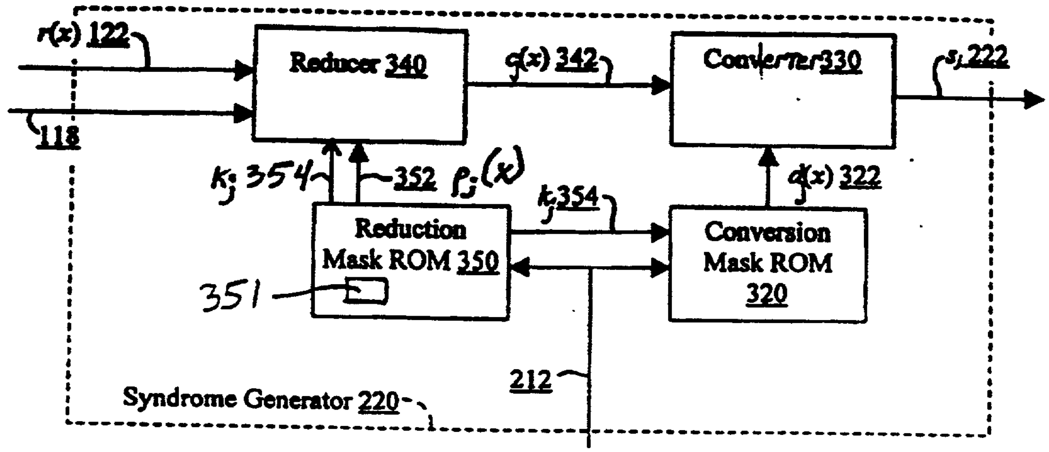

r(x) 122
118
Reducer 340
q(x) 342
Converter 330
s 222
k 354
352 p(x)
k 354
d(x) 322
Reduction
Mask ROM 350
351
Conversion
Mask ROM
320
212
Syndrome Generator 220

FIG. 3

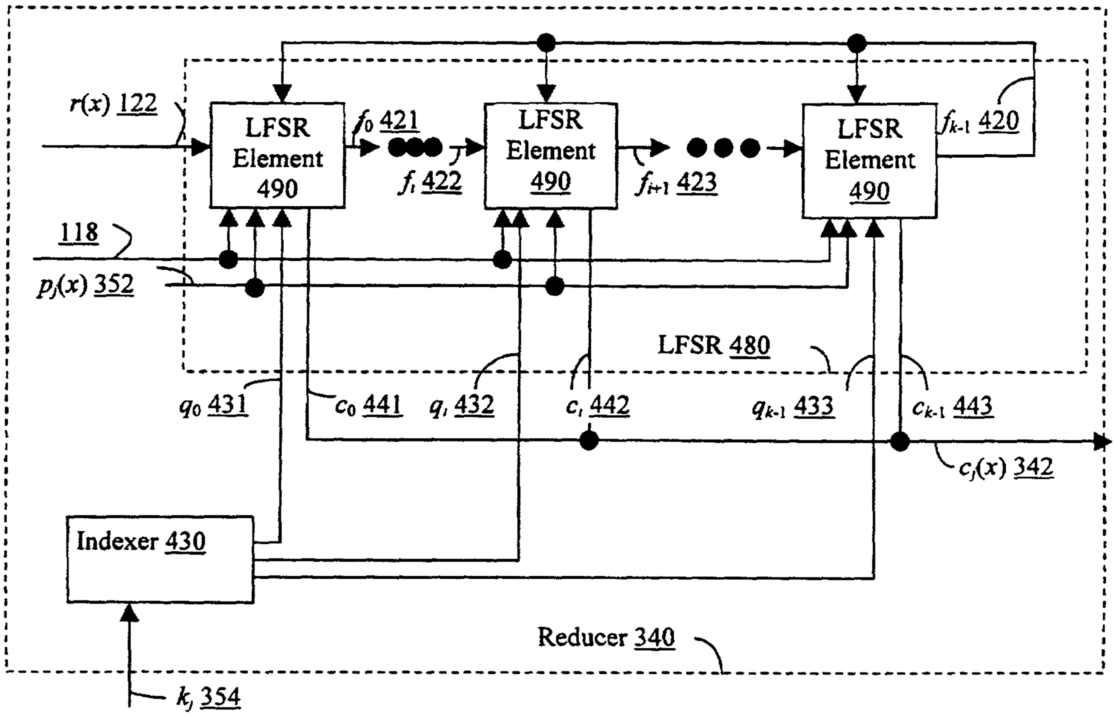
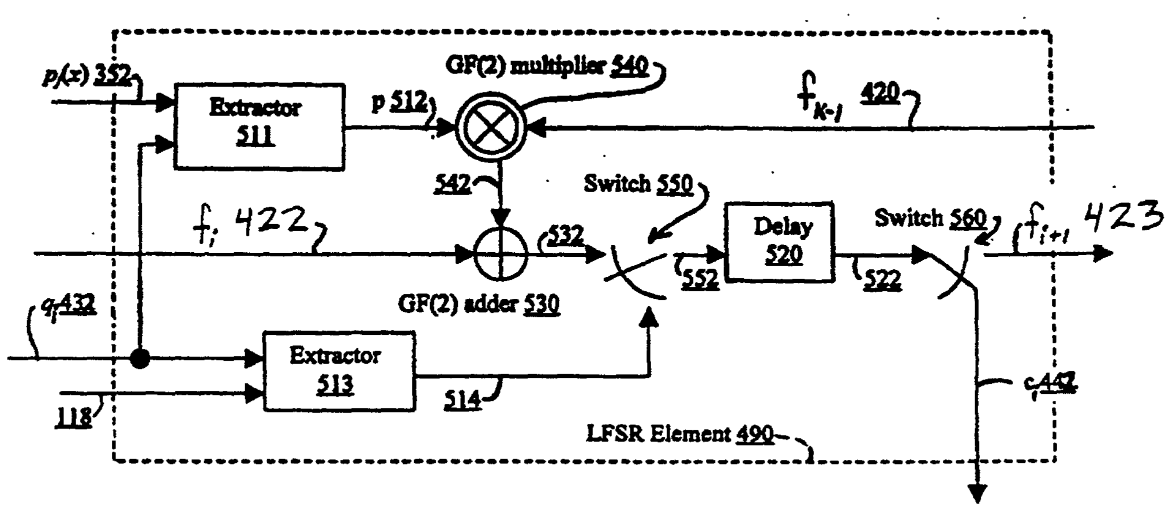

r(x) 122
LFSR Element 490
f0 421
fi 422
LFSR Element 490
fi+1 423
LFSR Element 490
fk-1 420
118
pj(x) 352
LFSR 480
q0 431
c0 441
qi 432
ci 442
qk-1 433
ck-1 443
cj(x) 342
Indexer 430
Reducer 340
kj 354

FIG. 4



p(x) 352
Extractor
511
p 512
GF(2) multiplier 540
f_{k-1} 420
542
Switch 550
f_i 422
532
Delay
520
Switch 560
f_{i+1} 423
552
522
GF(2) adder 530
q_i 432
Extractor
513
118
514
c_i 442
LFSR Element 490

FIG. 5

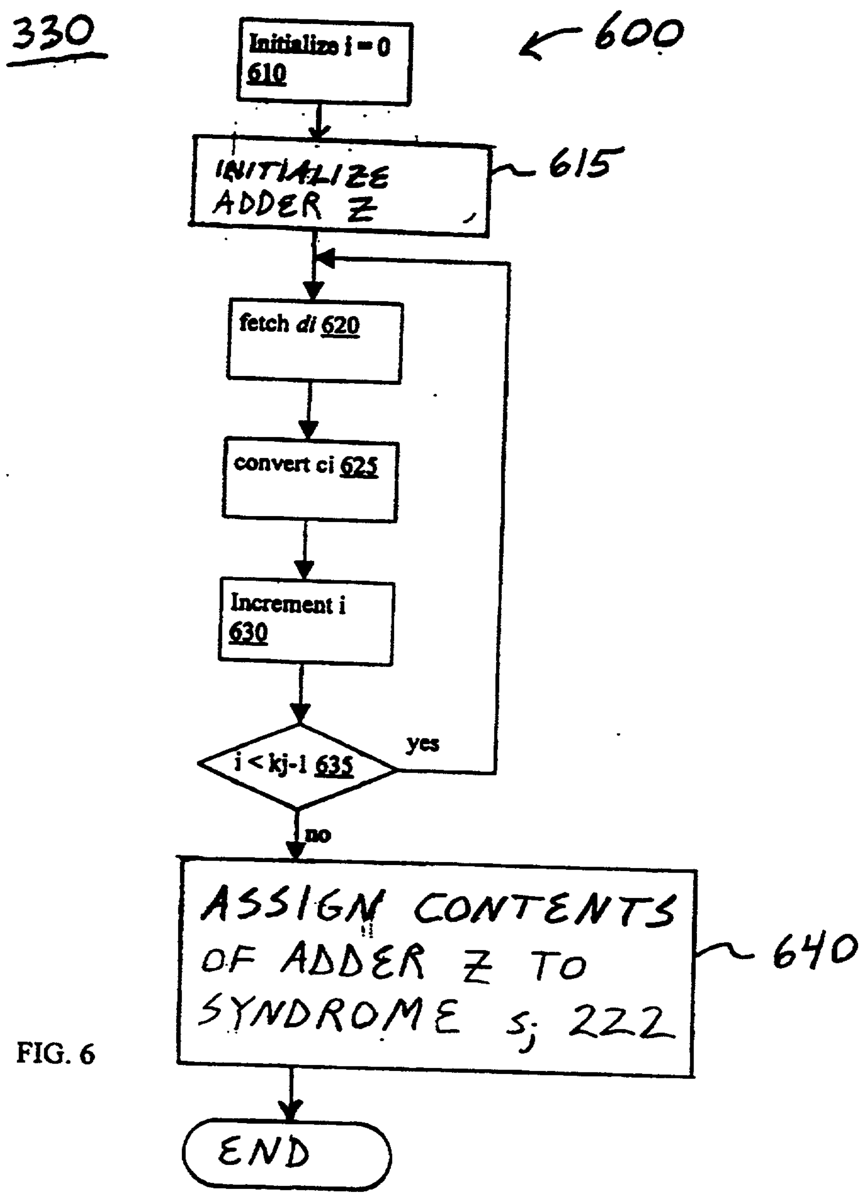

330
600
Initialize i = 0 610
INITIALIZE ADDER Z
615
fetch di 620
convert ci 625
Increment i 630
i < kj-1 635
yes
no
ASSIGN CONTENTS OF ADDER Z TO SYNDROME s_j 222
640
END

FIG. 6

240

700

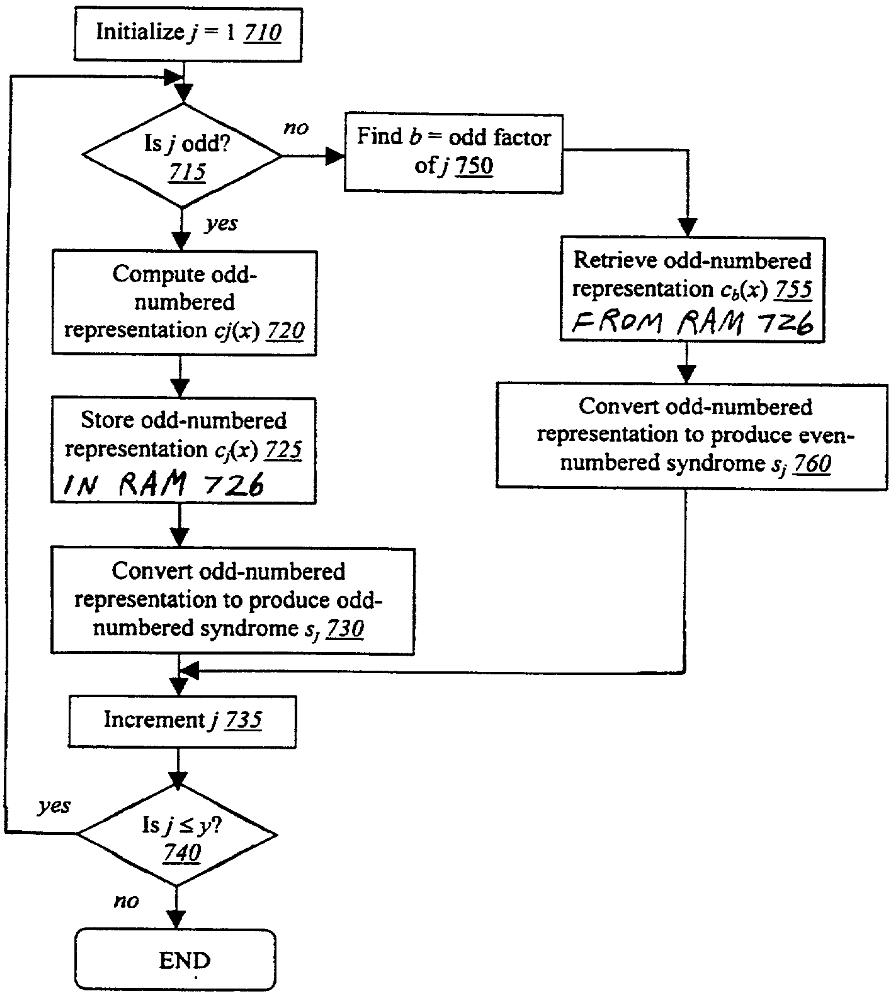

Initialize j = 1 710
Is j odd? 715
no
Find b = odd factor of j 750
yes
Compute odd-numbered representation $cj(x)$ 720
Retrieve odd-numbered representation $c_b(x)$ 755 FROM RAM 726
Store odd-numbered representation $c_j(x)$ 725 IN RAM 726
Convert odd-numbered representation to produce even-numbered syndrome s_j 760
Convert odd-numbered representation to produce odd-numbered syndrome s_j 730
Increment j 735
yes
Is $j \leq y$? 740
no
END

FIG. 7

PATENT APPLICATION DECLARATION COMBINED
WITH POWER OF ATTORNEY

__X__ REGULAR (UTILITY) OR ___ DESIGN APPLICATION
(check one)

Attorney Docket
No. __CR00219M__

As a below-named inventor, I hereby declare that:

My residence, post office address and citizenship are as stated below next to my name.

I believe I am the original, first and sole inventor (if only one name is listed below) or an original, first and joint inventor (if plural names are listed below) of the subject matter which is claimed and for which a patent is sought on the invention entitled: **DECODER-USABLE SYNDROME GENERATION WITH REPRESENTATION GENERATED WITH INFORMATION BASED ON VECTOR PORTION**, the specification of which:

(check one)

__X__ is attached hereto.

____ was filed on ______________________ as
U.S. Application Serial No. ______________
and was amended on ______________.
(if applicable)

I hereby state that I have reviewed and understand the contents of the above-identified specification, including the claims, as amended by any amendment referred to above.

I acknowledge the duty to disclose information which is material to the patentability of this application in accordance with Title 37, Code of Federal Regulations, Section 1.56(a).

I hereby claim foreign priority benefits under Title 35, United States Code, Section 119 of any foreign application(s) for patent or inventor's certificate listed below and have also identified below any foreign application for patent or inventor's certificate having a filing date before that of the application on which priority is claimed:

Prior Foreign Application(s):

(check one)

__X__ no such applications filed.

____ such applications identified as follows:

			Priority Claimed	
(Serial No.)	(Country)	(Day/Month/Year Filed)	Yes	No
(Serial No.)	(Country)	(Day/Month/Year Filed)	Yes	No
(Serial No.)	(Country)	(Day/Month/Year Filed)	Yes	No

I hereby claim the priority benefit under Title 35, United States Code, Section 120 of any United States application(s) listed below and, insofar as the subject matter of each of the claims of this application is not

disclosed in the prior United States application in the manner provided by the first paragraph of Title 35, United States Code, Section 112, I acknowledge the duty to disclose material information as defined in Title 37, Code of Federal Regulations, Section 1.56(a) which is material to the examination of this application and which occurred between the filing date of the prior application and the national or PCT international filing date of this application:

Prior U.S. Applications(s):

(check one)   __X__ no such applications filed.

____ such applications identified as follows:

(Application Serial No.)	(Filing Date)	(Status) (Patented, Pending, Abandoned)
(Application Serial No.)	(Filing Date)	(Status) (Patented, Pending, Abandoned)
(Application Serial No.)	(Filing Date)	(Status) (Patented, Pending, Abandoned)

I hereby declare that: as to any claimed subject matter of this application which is common to my earlier United States or foreign application(s), if any, which I have identified above and claimed the benefit of priority thereof, I do not believe that the same was ever known or used in the United States before my invention thereof or patented or described in any printed publication in any country before my invention thereof or more than one year prior to the first of said earlier application(s), or in public use or on sale in the United States more than one year prior to the first of said earlier application(s), and that the said common subject matter has not been patented or made the subject of an inventor's certificate before the date of the first of said earlier U.S. application(s) in any country foreign to the United States on an application, filed by me or my legal representatives or assigns more than twelve months (six months if the present application is a Design patent application) prior to the first of said earlier U.S. application(s), if any; and that, as to any claimed subject matter of this application which is not common to said earlier application(s), if any, I do not know and do not believe that the same was ever known or used in the United States before my invention thereof or patented or described in any printed publication in any country before my invention thereof or more than one year prior to the date of this application, or in public use or on sale in the United States more than one year prior to the date of this application, and that said subject matter has not been patented or made the subject of an inventor's certificate in any country foreign to the United States on an application filed by me or my legal representatives or assigns more than twelve months (six months if the present application is a Design patent application) prior to the date of this application.

I HEREBY APPOINT THE ATTORNEY(S) OR AGENT(S) ASSOCIATED WITH:
CUSTOMER NUMBER 22917
TO PROSECUTE THIS APPLICATION AND TRANSACT ALL
BUSINESS IN THE PATENT AND TRADEMARK OFFICE CONNECTED THEREWITH.

Send Written Correspondence To:

Customer Number **22917**

Direct Telephone Calls to: (847) 576-6364

I hereby declare that all statements made herein of my own knowledge are true and that all statements made on information and belief are believed to be true; and further that these statements were made with the knowledge that willful false statement and the like so made are punishable by fine or imprisonment, or both, under Section 1001 of Title 18 of the United States Code and that such willful false statements may jeopardize the validity of the application or any patent issued thereon.

Full name of sole or first inventor: BRIAN KEITH CLASSON

Inventor's signature: Brian K Classon

Date:

Residence: STREAMWOOD (City) IL (State or Foreign Country)

Citizenship: USA

Post Office Address: 124 RED CEDAR DRIVE (Street Address)

STREAMWOOD (City) IL (State or Country) 60107 (Zip Code)

Full name of sole or first inventor: VIPUL ANIL DESAI

Inventor's signature: Vip Desai

Date:

Residence: HOFFMAN ESTATES (City) IL (State or Foreign Country)

Citizenship: USA

Post Office Address: 1480 HASSELL ROAD (Street Address)

HOFFMAN ESTATES (City) IL (State or Country) 60195 (Zip Code)